Ch:02

Architecture:

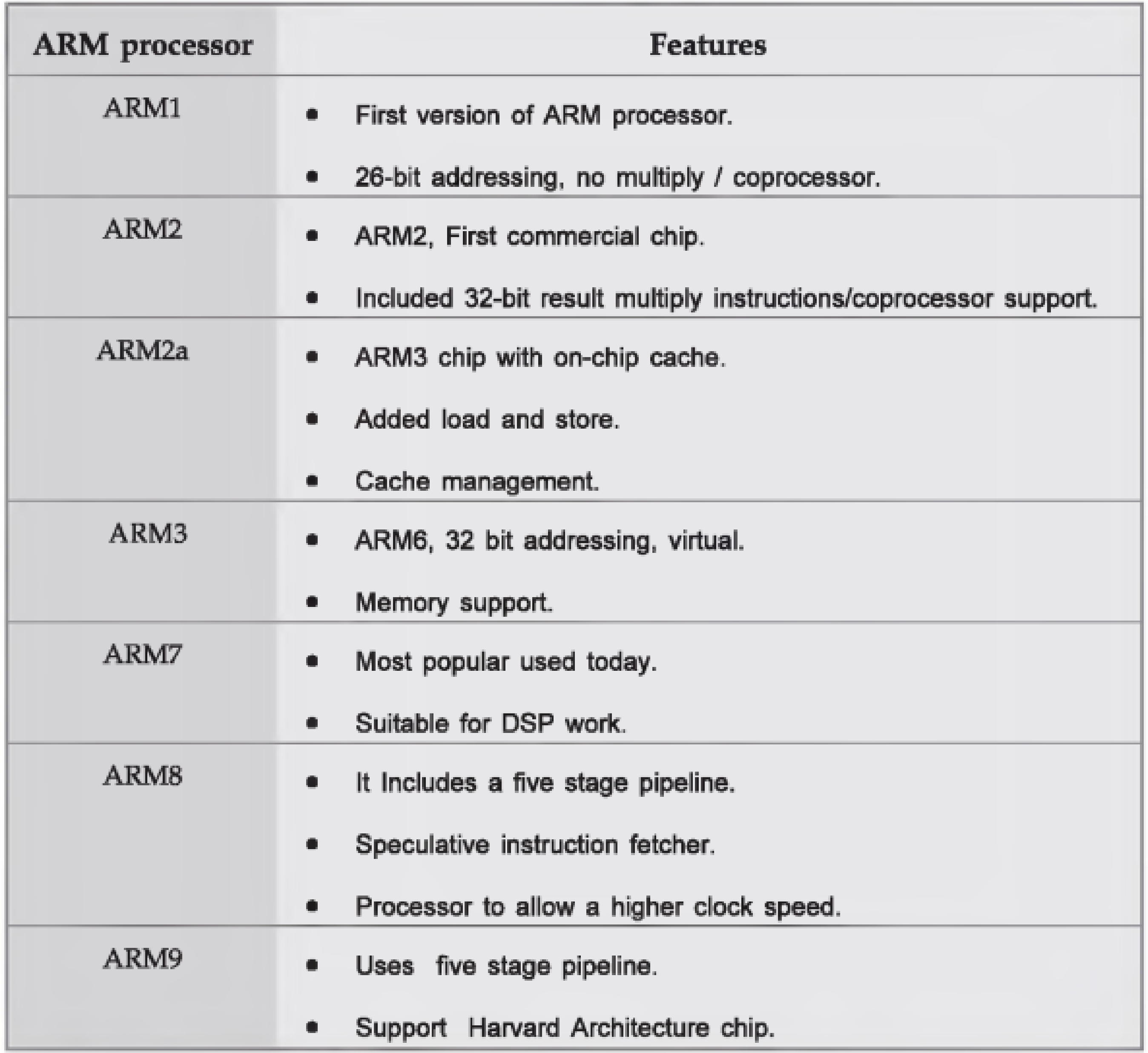
- Basic ARM architecture, ARM organization core Data flow Model,
- ARM register organization, current program register organization.
- Pin configuration and architecture Arduino,
- Introduction to Raspberry Pi, Understanding SoC architecture and SoCs used in
- Raspberry Pi, Pin Description of Raspberry Pi,
- On-board components of Rpi
- Microcontroller Applications: Interfacing matrix keyboard and
- Seven segments LED display,
- LCD Interfacing,
- ADC Interfacing,
- DC motor interfacing.

ARM Powered Products

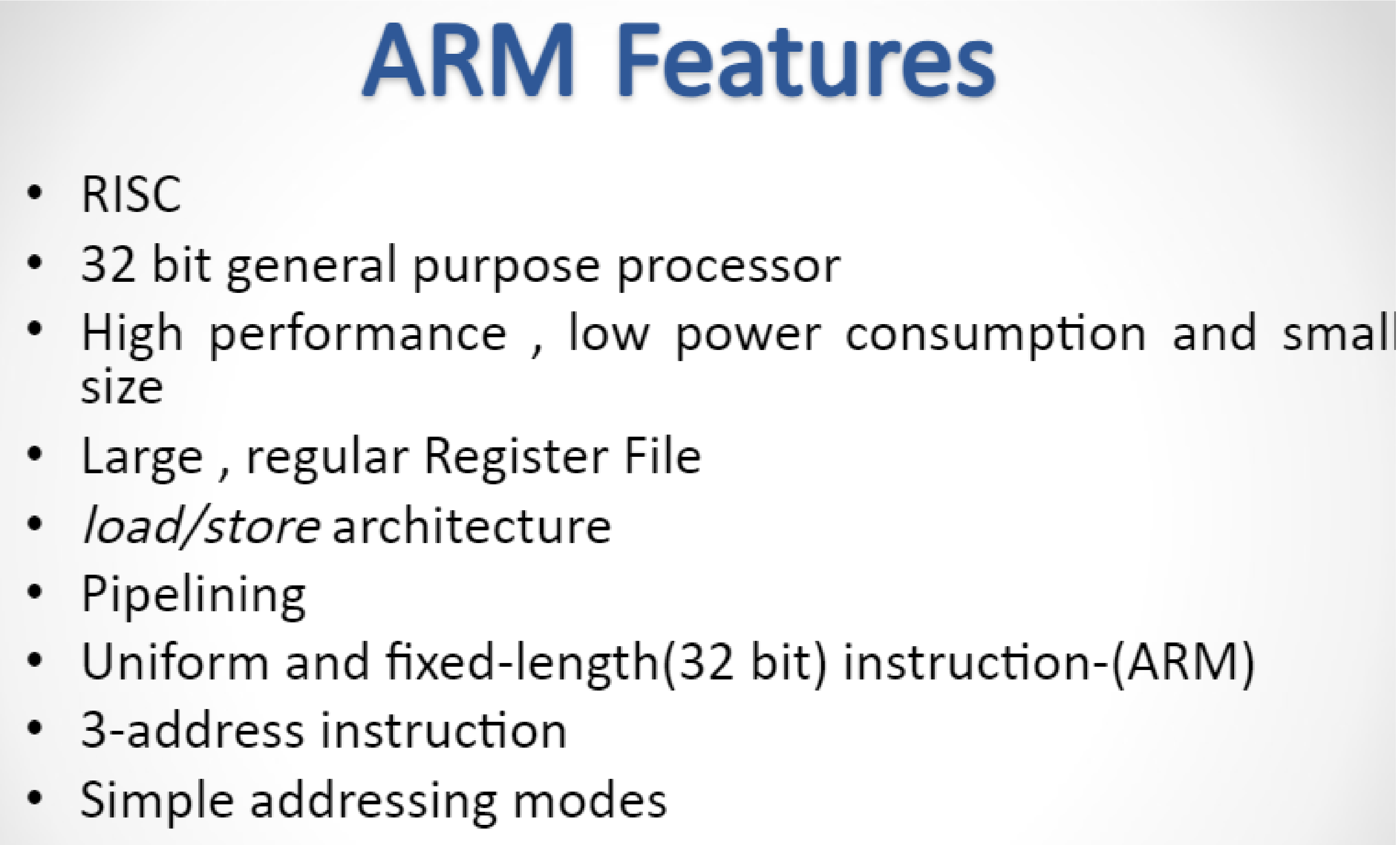
ARM Architecture

- **ARM Architecture Versions**
- ARM means Advanced RISC Machines. ARM machines have a 32-bit Reduced Instruction Set Computer (RISC) Load Store Architecture.
- It is first RISC microprocessor for commercial use and market-leader for low power and cost-sensitive embedded applications.
- The processor originated in England in 1984. At its inception ARM stood for Acorn RISC Machine.
- They were used mostly for British educational systems and therefore, were not widely available or known outside England. However in 1987 the ARM became the first commercial RISC processor.
- The ARM is a Von Neumann, load/store architecture i.e. only 32-bit data bus for both instruction and data. Also for the load/store instruction access memory.
- The ARM is a 32-bit architecture.
- When used in relation to the ARM : 1. Byte means 8 bits 2. Halfword means 16 bits. 3. Word means 32 bits.
- Most ARM's implement two instruction sets
- 1. 32-bit ARM Instruction Set
- 2. 16-bit Thumb Instruction Set(THUMB instruction set is optimized for smallest code size, so your program can fit into the smallest amount of storage (e.g. flash) possible.)

- Memory is addressed as a 32 bit address space.
- ARM1 processor was the first commercialized RISC processor and it contained 25,000 transistors.

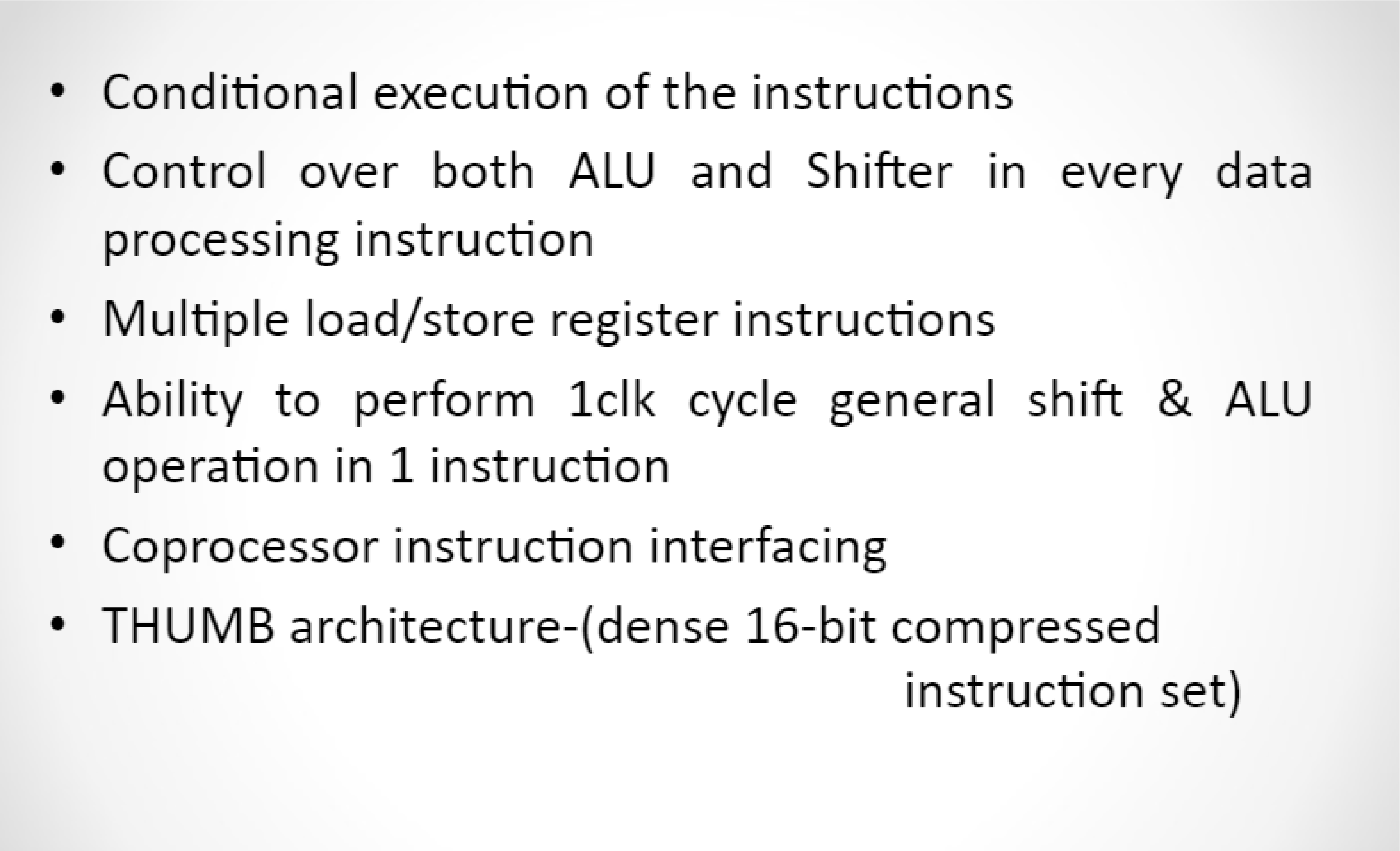


ARM processor	Features
ARM1	● First version of ARM processor. ● 26-bit addressing, no multiply / coprocessor.
ARM2	● ARM2, First commercial chip. ● Included 32-bit result multiply instructions/coprocessor support.
ARM2a	● ARM3 chip with on-chip cache. ● Added load and store. ● Cache management.
ARM3	● ARM6, 32 bit addressing, virtual. ● Memory support.
ARM7	● Most popular used today. ● Suitable for DSP work.
ARM8	● It Includes a five stage pipeline. ● Speculative instruction fetcher. ● Processor to allow a higher clock speed.
ARM9	● Uses five stage pipeline. ● Support Harvard Architecture chip.



ARM Features

- RISC
- 32 bit general purpose processor
- High performance , low power consumption and small size
- Large , regular Register File
- *load/store* architecture
- Pipelining
- Uniform and fixed-length(32 bit) instruction-(ARM)
- 3-address instruction
- Simple addressing modes



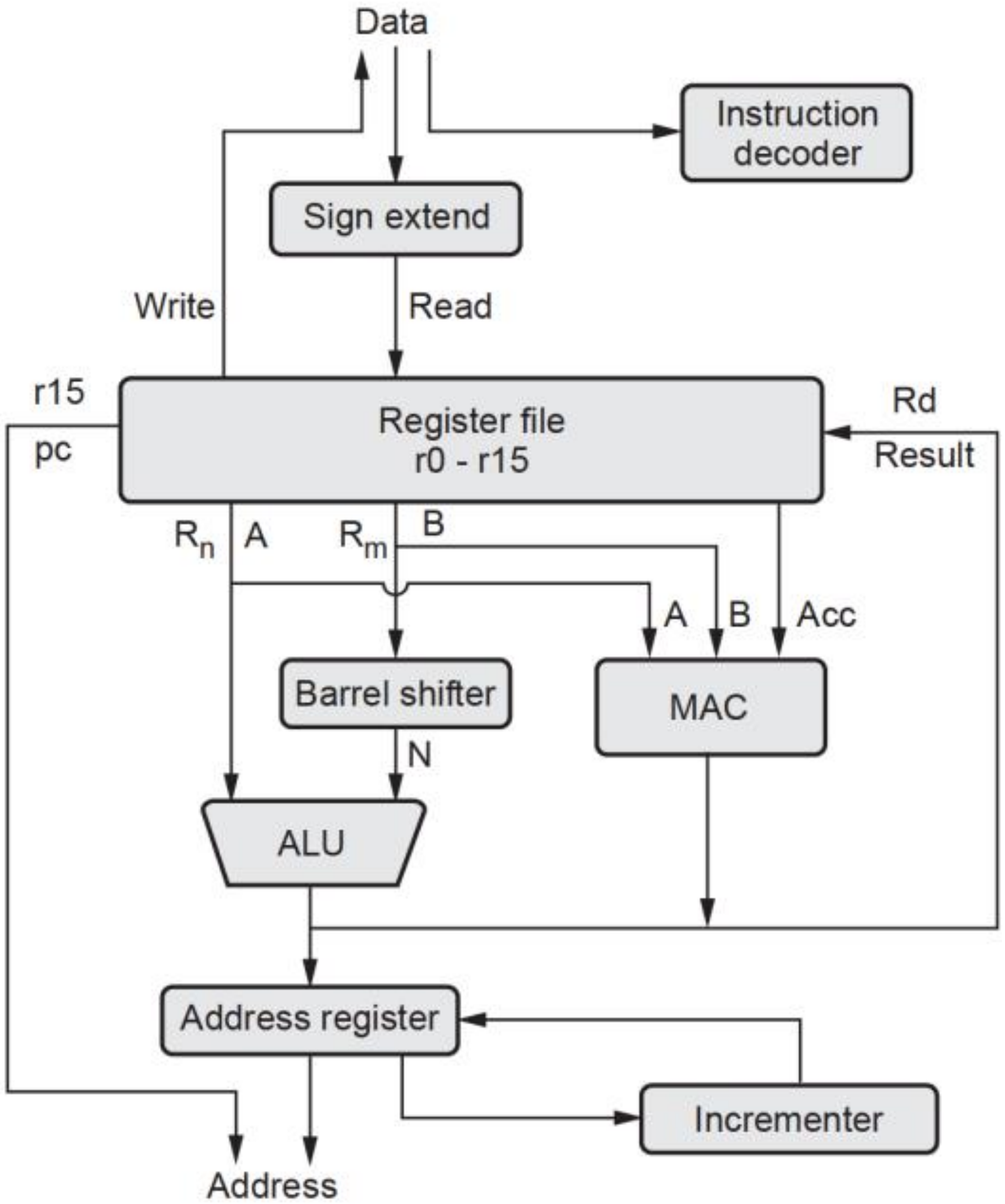
- Conditional execution of the instructions
- Control over both ALU and Shifter in every data processing instruction
- Multiple load/store register instructions
- Ability to perform 1clk cycle general shift & ALU operation in 1 instruction
- Coprocessor instruction interfacing
- THUMB architecture-(dense 16-bit compressed instruction set)

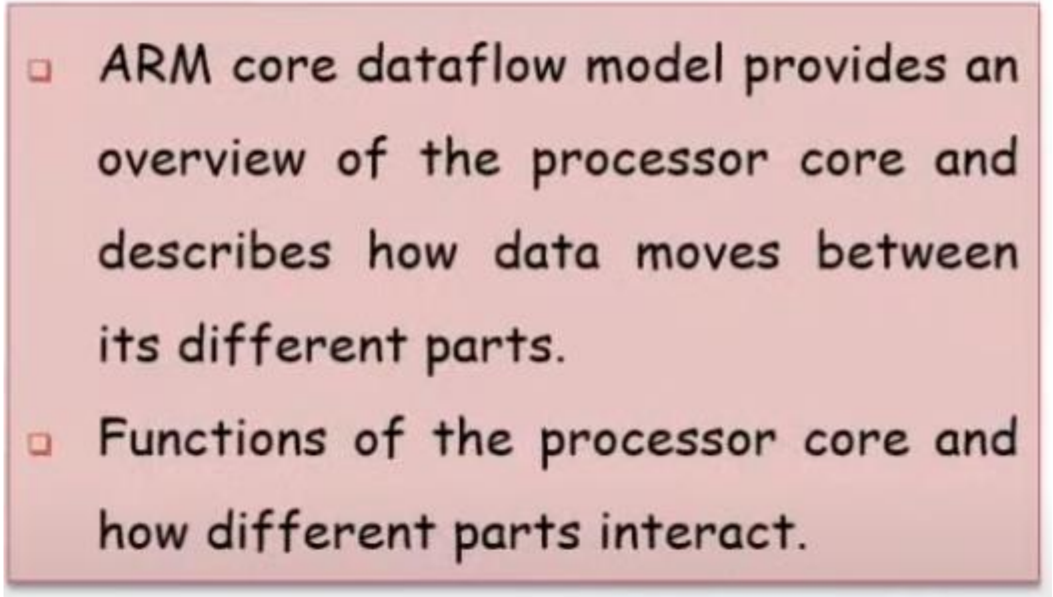
ARM Architecture

- The ARM architecture processor is an advanced reduced instruction set computing [RISC] machine and it's a 32 bit RISC microcontroller.

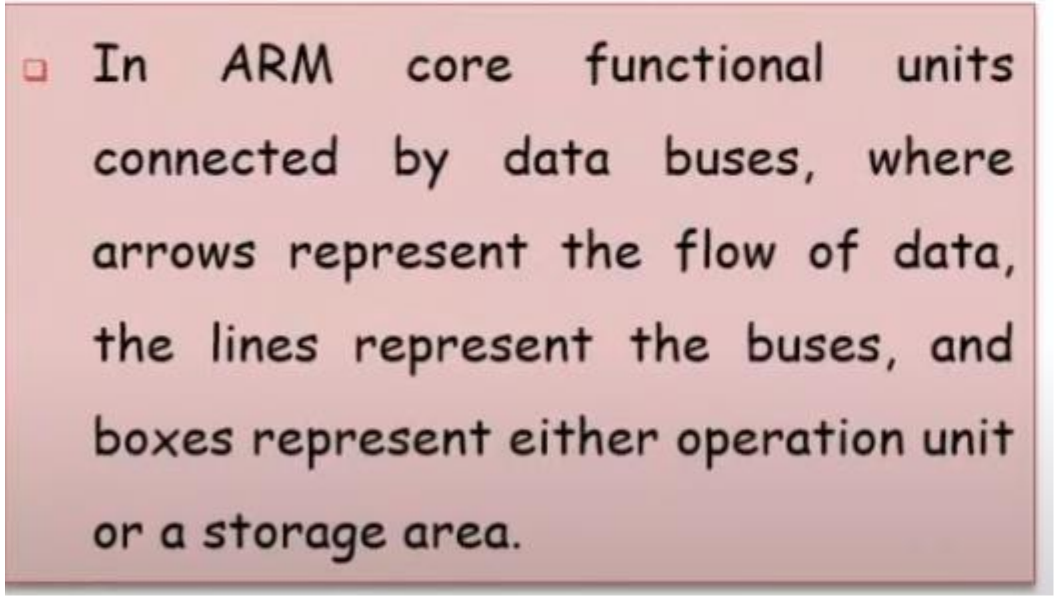
- The ARM Architecture consists of following:
  - a) Arithmetic Logic Unit
  - b) Booth multiplier
  - c) Barrel shifter
  - d) Control unit
  - e) Register file

ARM Architecture

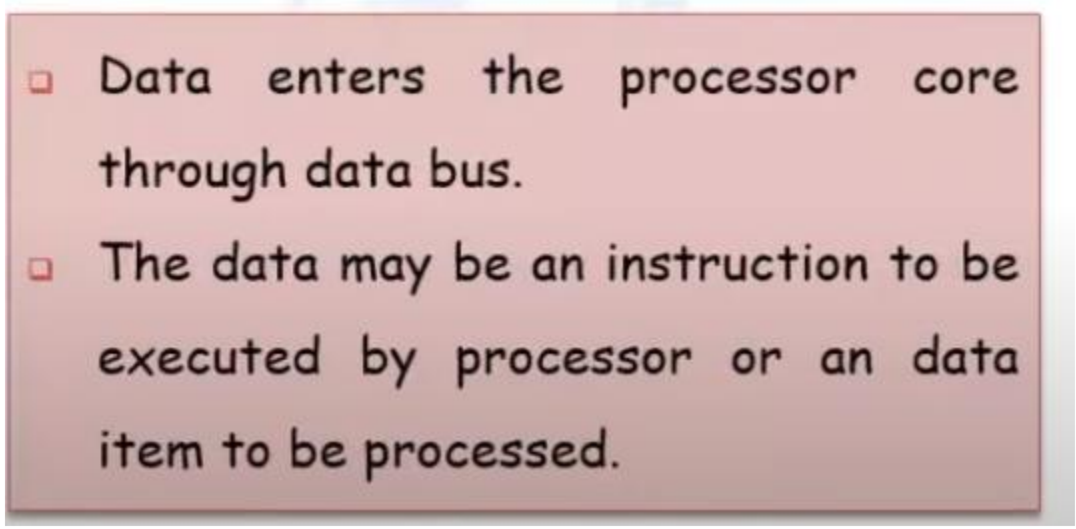




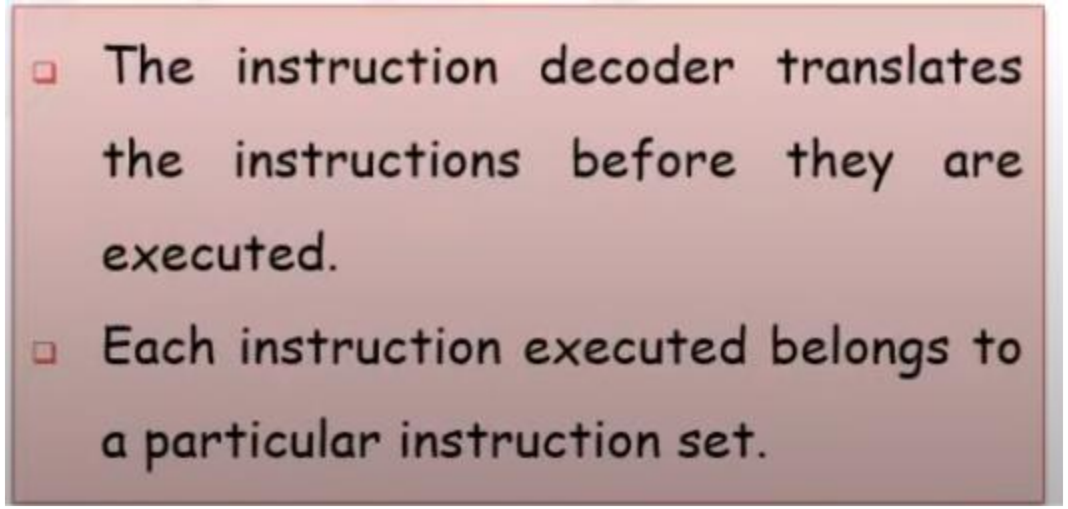
- ARM core dataflow model provides an overview of the processor core and describes how data moves between its different parts.
- Functions of the processor core and how different parts interact.



- In ARM core functional units connected by data buses, where arrows represent the flow of data, the lines represent the buses, and boxes represent either operation unit or a storage area.



- Data enters the processor core through data bus.
- The data may be an instruction to be executed by processor or an data item to be processed.

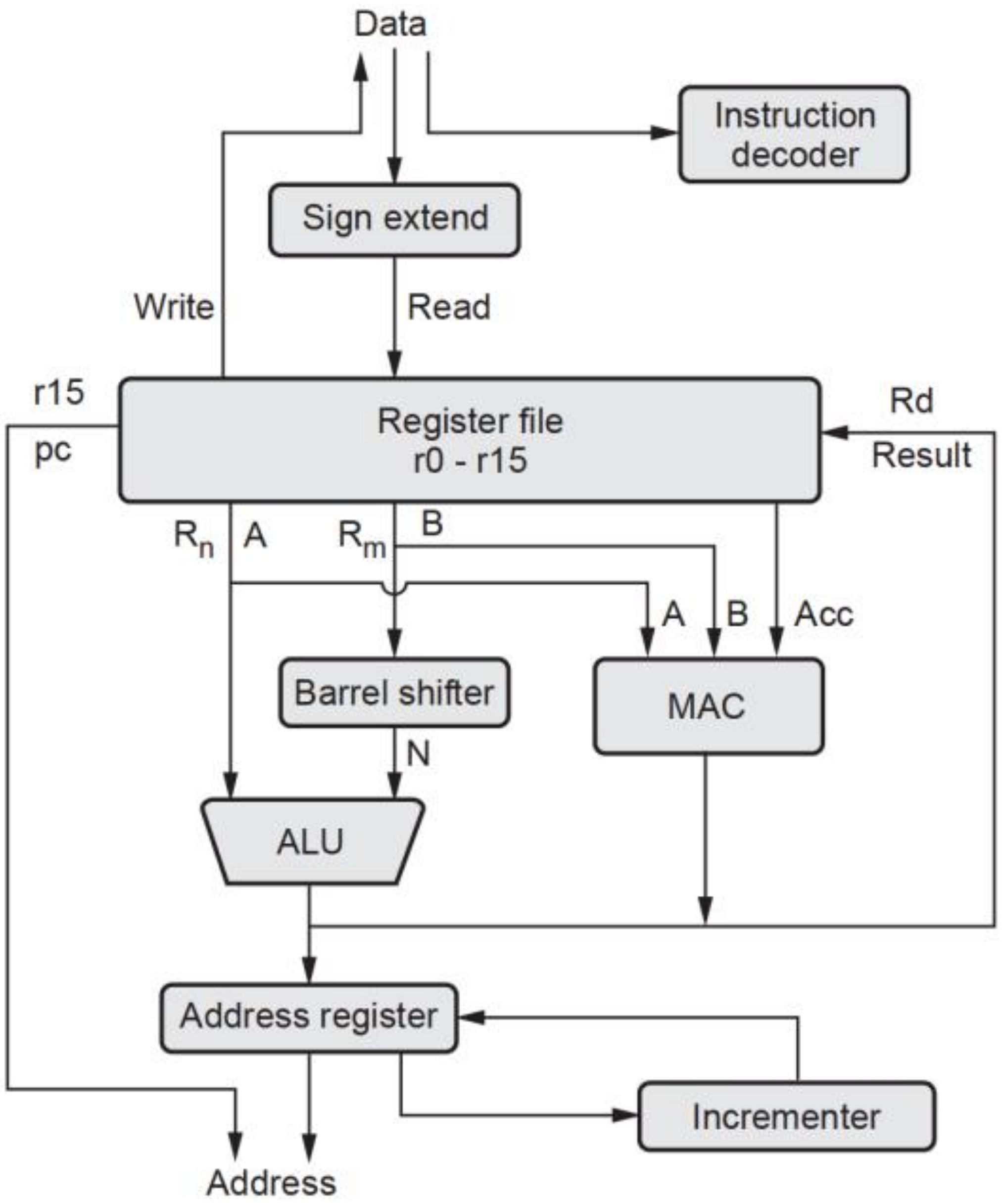


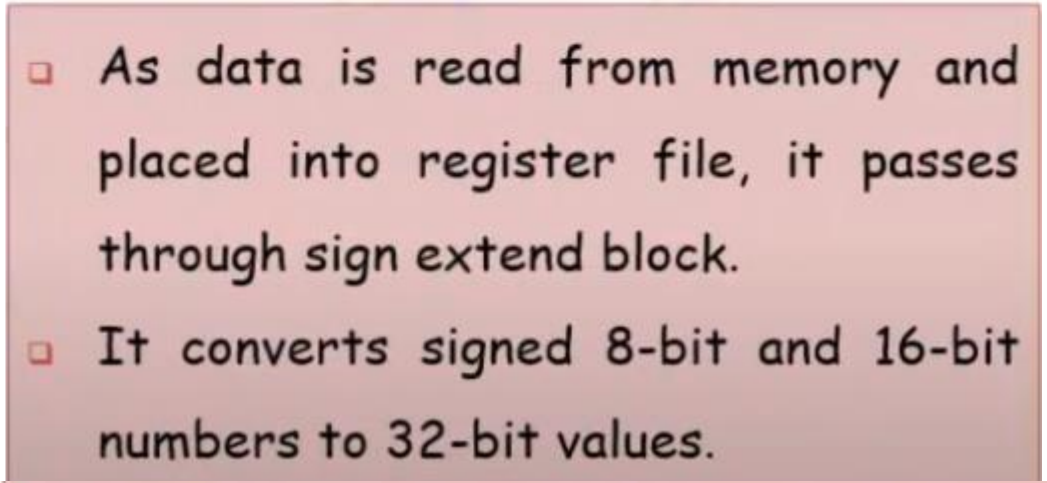
- The instruction decoder translates the instructions before they are executed.
- Each instruction executed belongs to a particular instruction set.

- It uses Load-Store architecture means, it has two instruction types for transferring data in and out of the processor
- Load instructions copy data from memory to registers in the core
- Store instructions copy data from registers to memory.

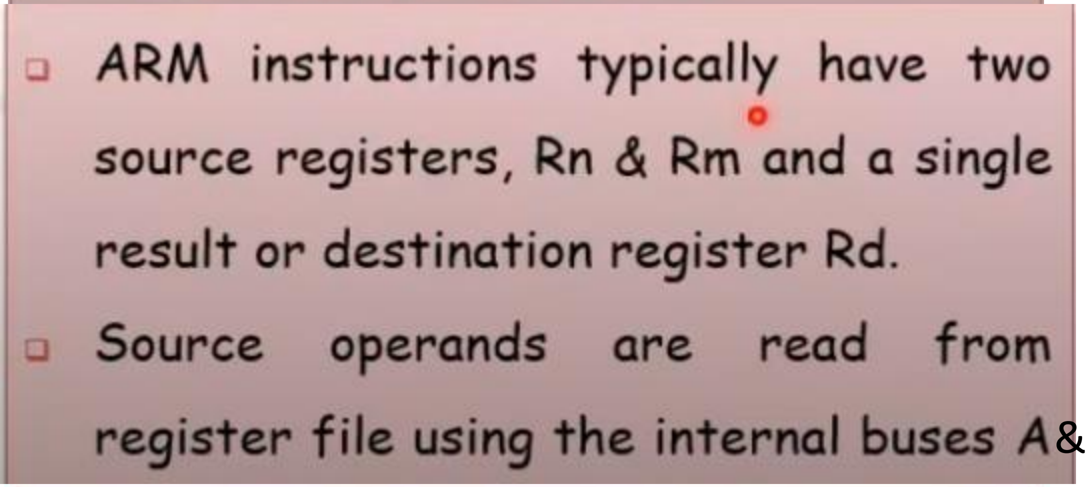
- Data items are placed in register file a storage bank made up of 32-bit registers.
- Since ARM core is 32-bit processor, most instructions treat the registers as holding signed or unsigned 32-bit values.

ARM Architecture

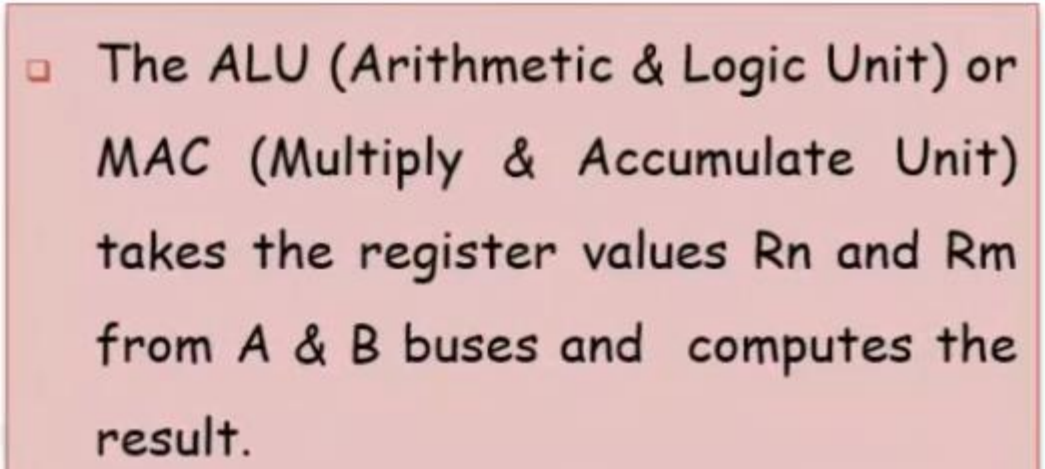




- As data is read from memory and placed into register file, it passes through sign extend block.
- It converts signed 8-bit and 16-bit numbers to 32-bit values.

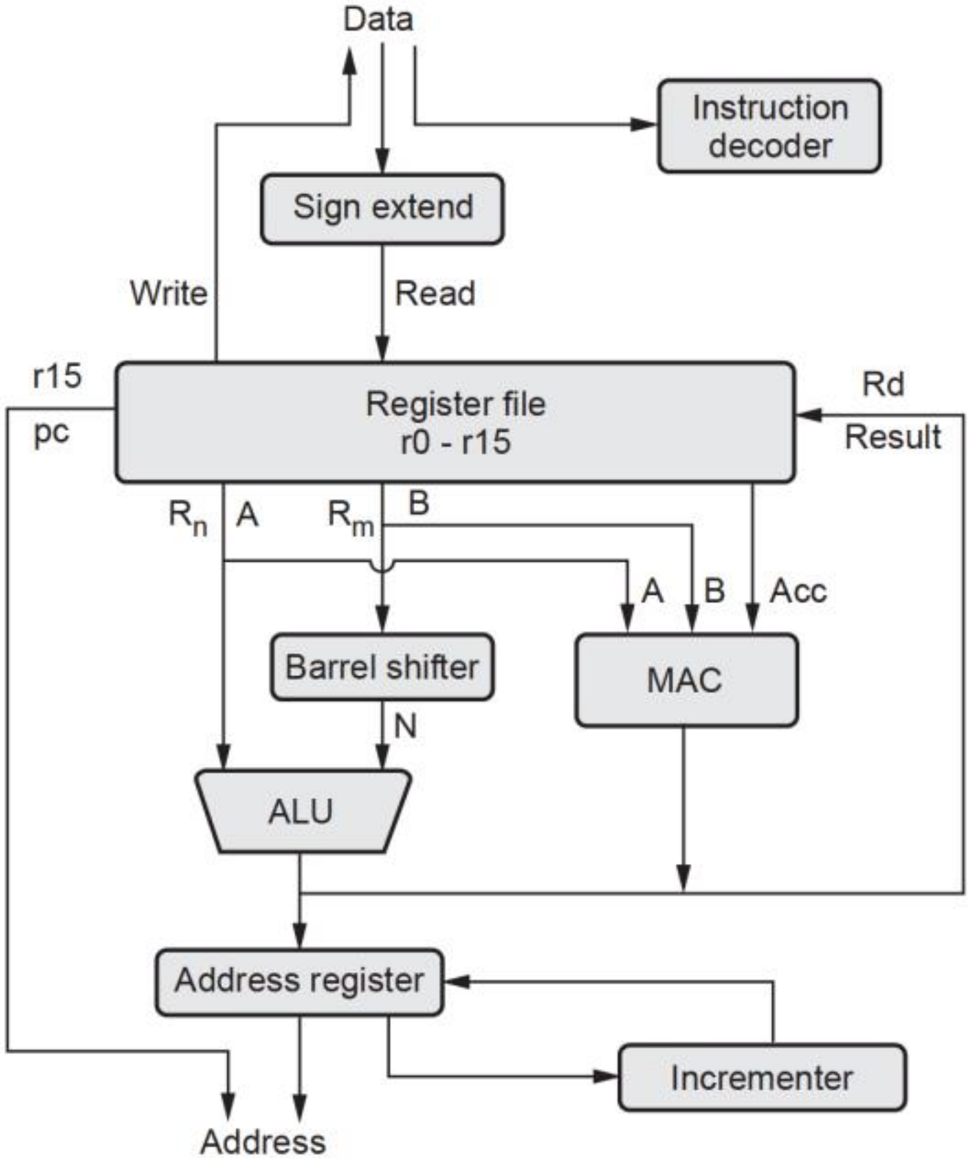


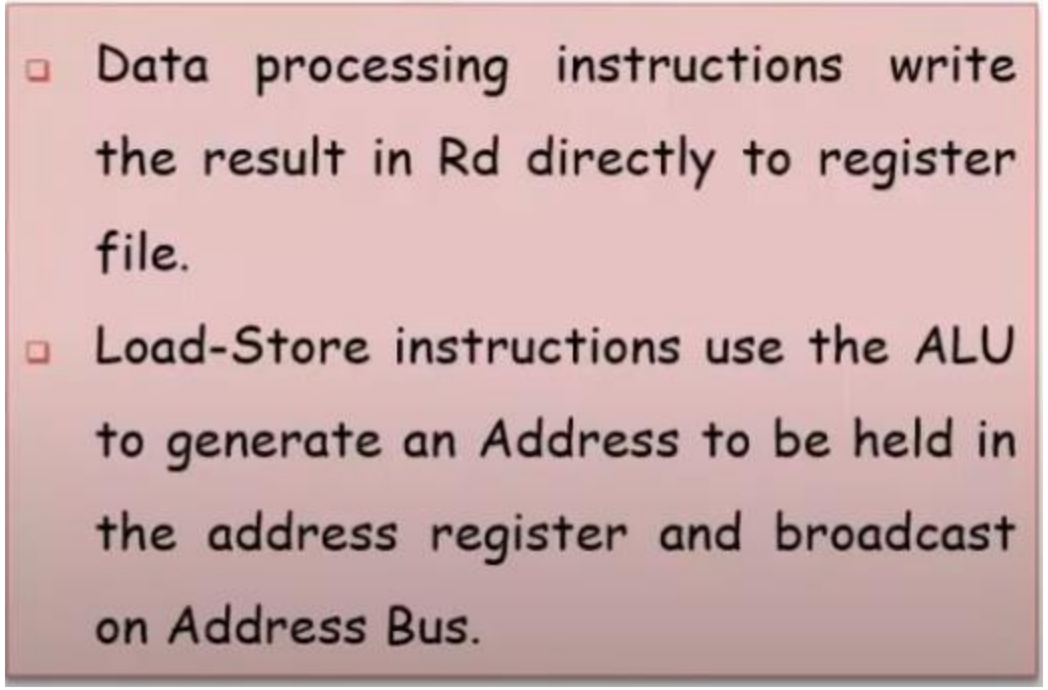
- ARM instructions typically have two source registers, Rn & Rm and a single result or destination register Rd.
- Source operands are read from register file using the internal buses A& B



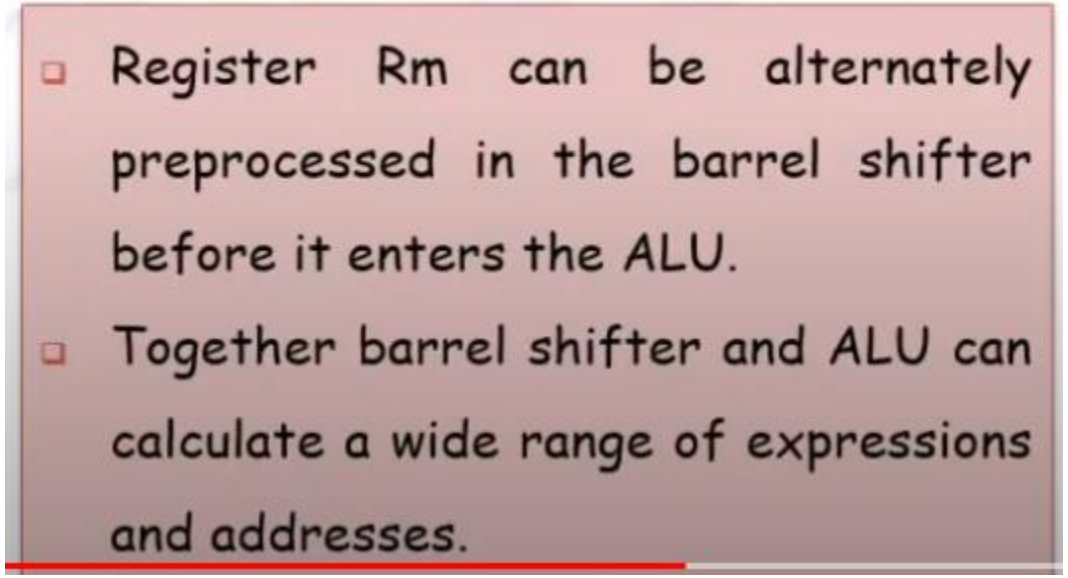
- The ALU (Arithmetic & Logic Unit) or MAC (Multiply & Accumulate Unit) takes the register values Rn and Rm from A & B buses and computes the result.

ARM Architecture





- Data processing instructions write the result in Rd directly to register file.
- Load-Store instructions use the ALU to generate an Address to be held in the address register and broadcast on Address Bus.



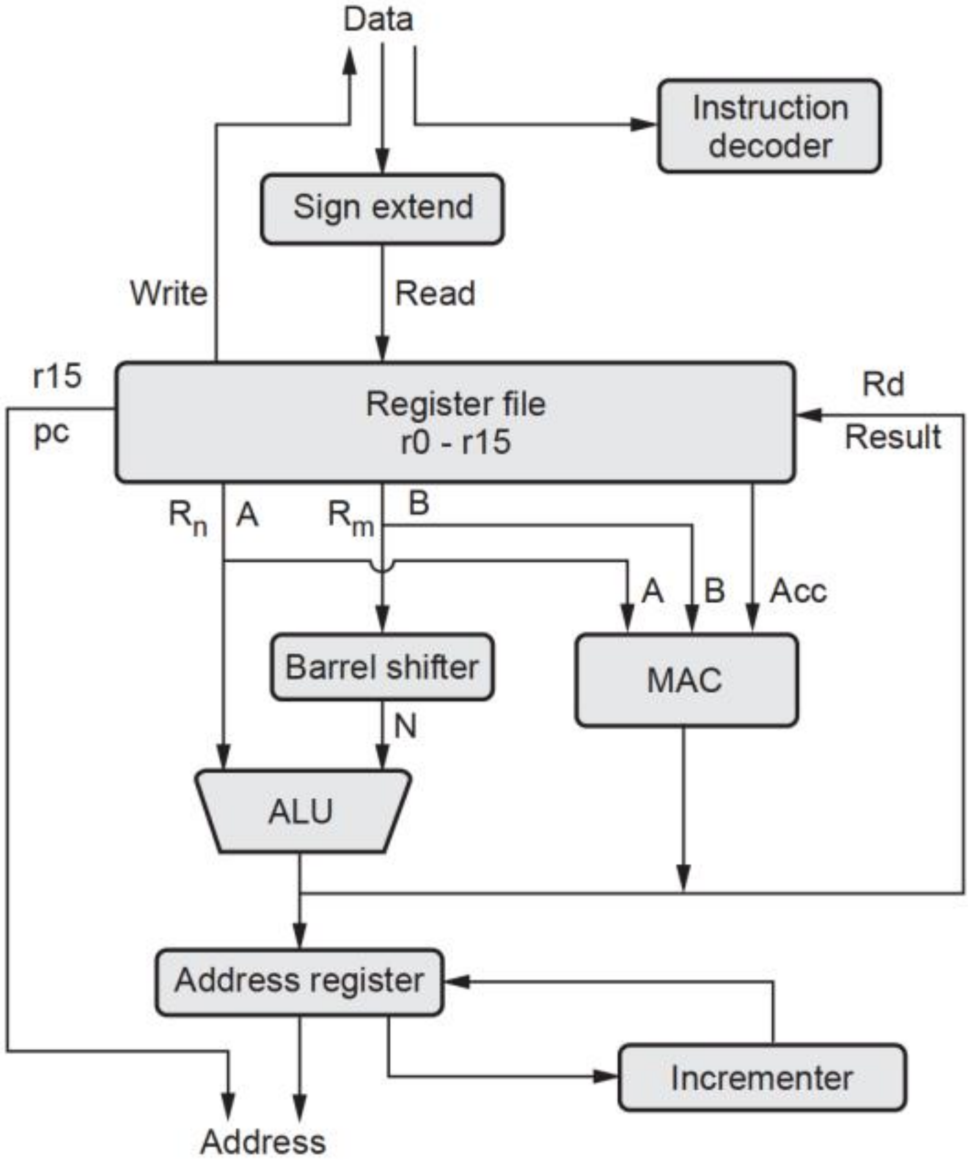
- Register Rm can be alternately preprocessed in the barrel shifter before it enters the ALU.
- Together barrel shifter and ALU can calculate a wide range of expressions and addresses.

Barrel shifter is a kind of support that is very useful in association with ALU for expression evaluation and address calculation.

- For load and store instructions incrementor updates address register before the core reads or writes the next register value from or to the next sequential memory location.

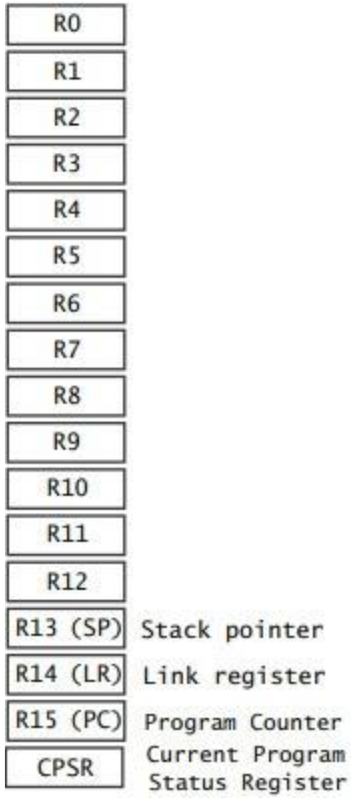
- The processor continues executing instructions until an exception or interrupt changes the normal execution flow.

ARM Architecture



- Address Register :
  - This holds the address generated by the load and store instructions and places it on the address bus.
- Instruction decoder :
  - It decodes the instruction opcode read from the memory and then the instruction is executed.
- Register file :
  - This is a bank of 32-bit registers used for storing data items.

ARM Register Organization

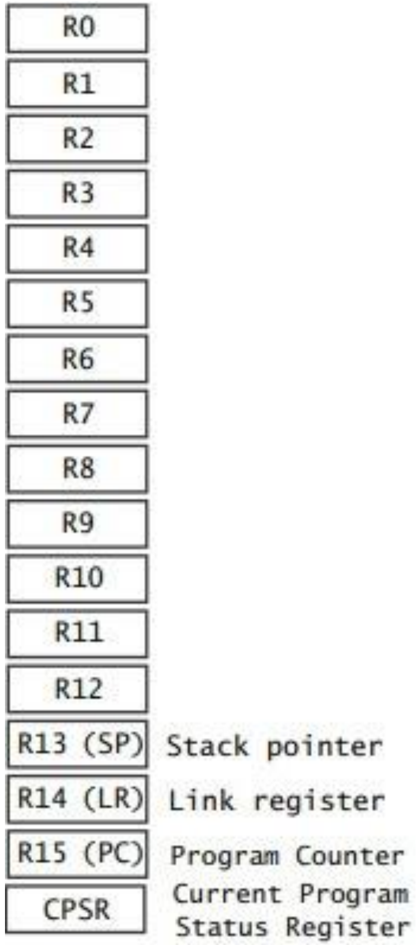


ARM Registers

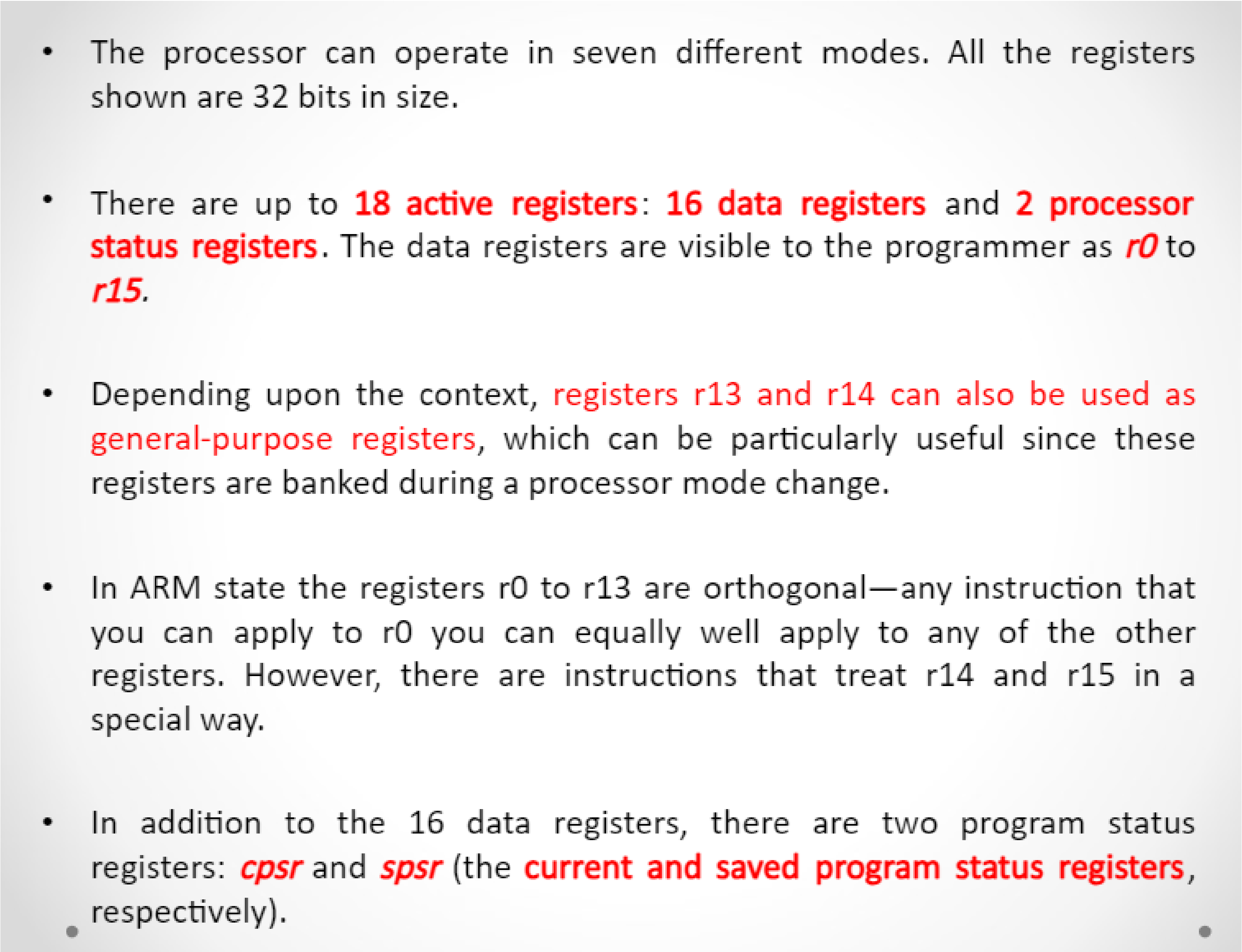
- General-purpose registers hold either data or an address.
- Fig 2 shows the active registers available in user mode—a protected mode normally used when executing applications.

- The ARM processor has three registers assigned to a particular task or special function: *r13*, *r14*, and *r15*. They are frequently given different labels to differentiate them from the other registers.

- In Fig 2, the shaded registers identify the assigned special-purpose registers:
- **Register *r13*** is traditionally used as the **stack pointer (sp)** and stores the head of the stack in the current processor mode.
- **Register r14** is called the **link register (*lr*)** and is where the core puts the return address whenever it calls a subroutine.
- **Register *r15*** is the **program counter (pc)** and contains the address of the next instruction to be fetched by the processor.



R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (SP) Stack pointer
R14 (LR) Link register
R15 (PC) Program Counter
CPSR Current Program Status Register



- The processor can operate in seven different modes. All the registers shown are 32 bits in size.

- There are up to **18 active registers**: **16 data registers** and **2 processor status registers**. The data registers are visible to the programmer as ***r0*** to ***r15***.

- Depending upon the context, registers r13 and r14 can also be used as general-purpose registers, which can be particularly useful since these registers are banked during a processor mode change.

- In ARM state the registers r0 to r13 are orthogonal—any instruction that you can apply to r0 you can equally well apply to any of the other registers. However, there are instructions that treat r14 and r15 in a special way.

- In addition to the 16 data registers, there are two program status registers: ***cpsr*** and ***spsr*** (the **current and saved program status registers**, respectively).

CPSR (Current Program Status Register)

- The ARM core uses the *cpsr* to monitor and control internal operations.

- The *cpsr* is a dedicated 32-bit register and resides in the register file.

- The *cpsr* is divided into **four fields, each 8 bits wide**: **flags**, status, extension, and control. In current designs the extension and status fields are reserved for future use. The control field contains the processor mode, state, and interrupt mask bits. The flags field contains the condition flags.

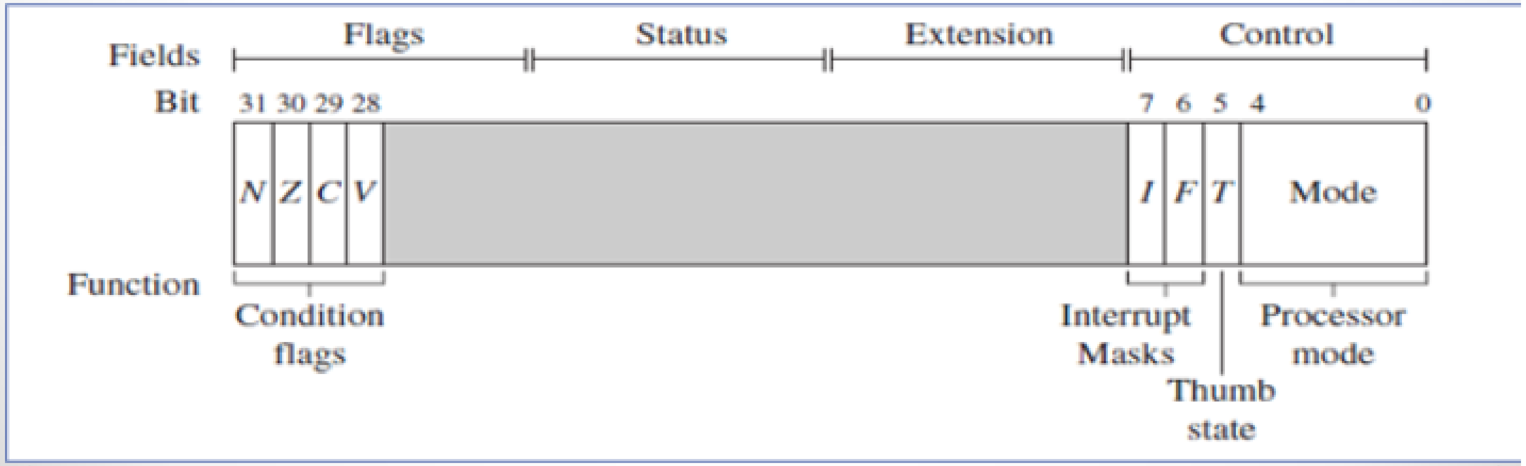
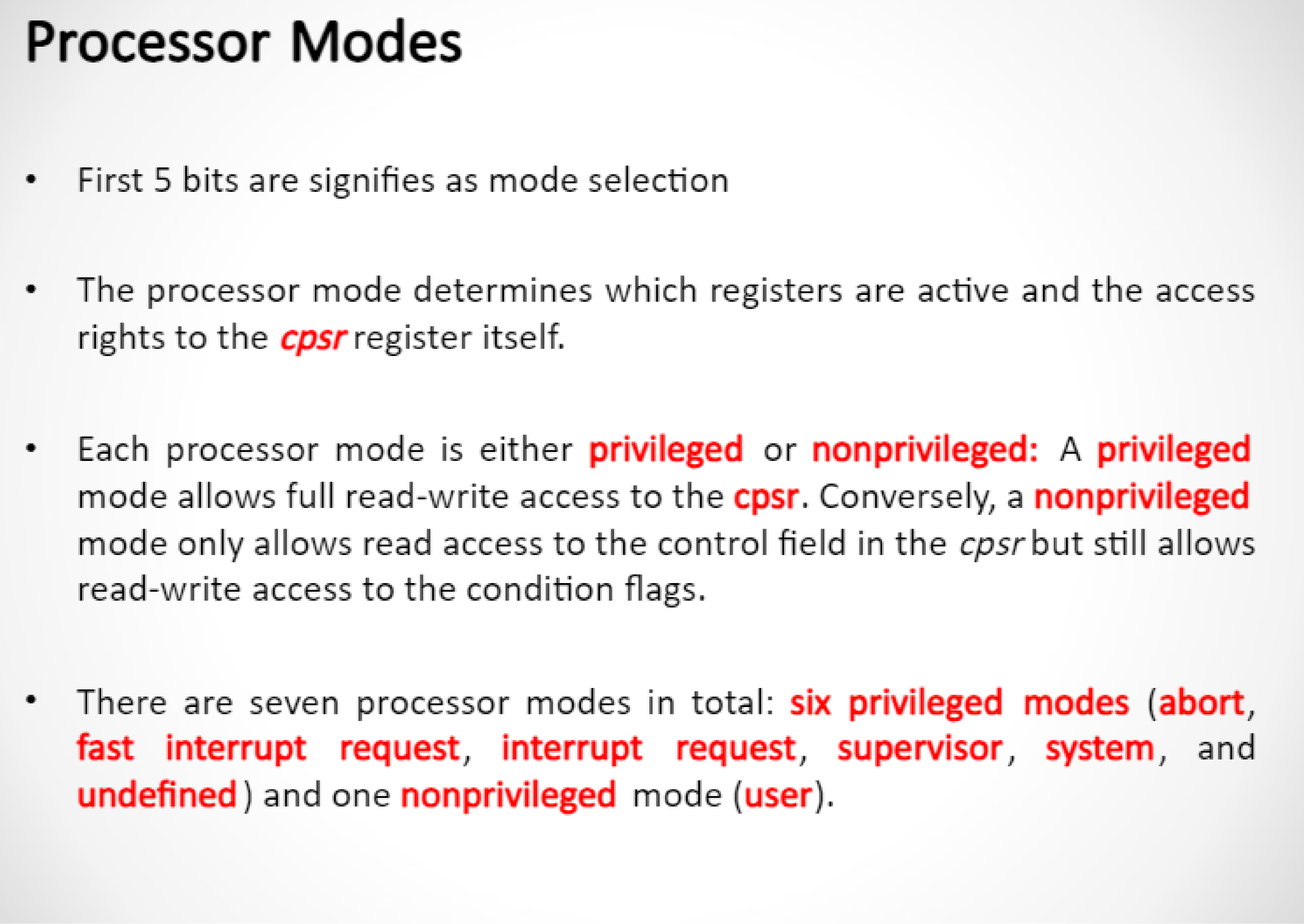


Fig 3: A generic program status register (*psr*).



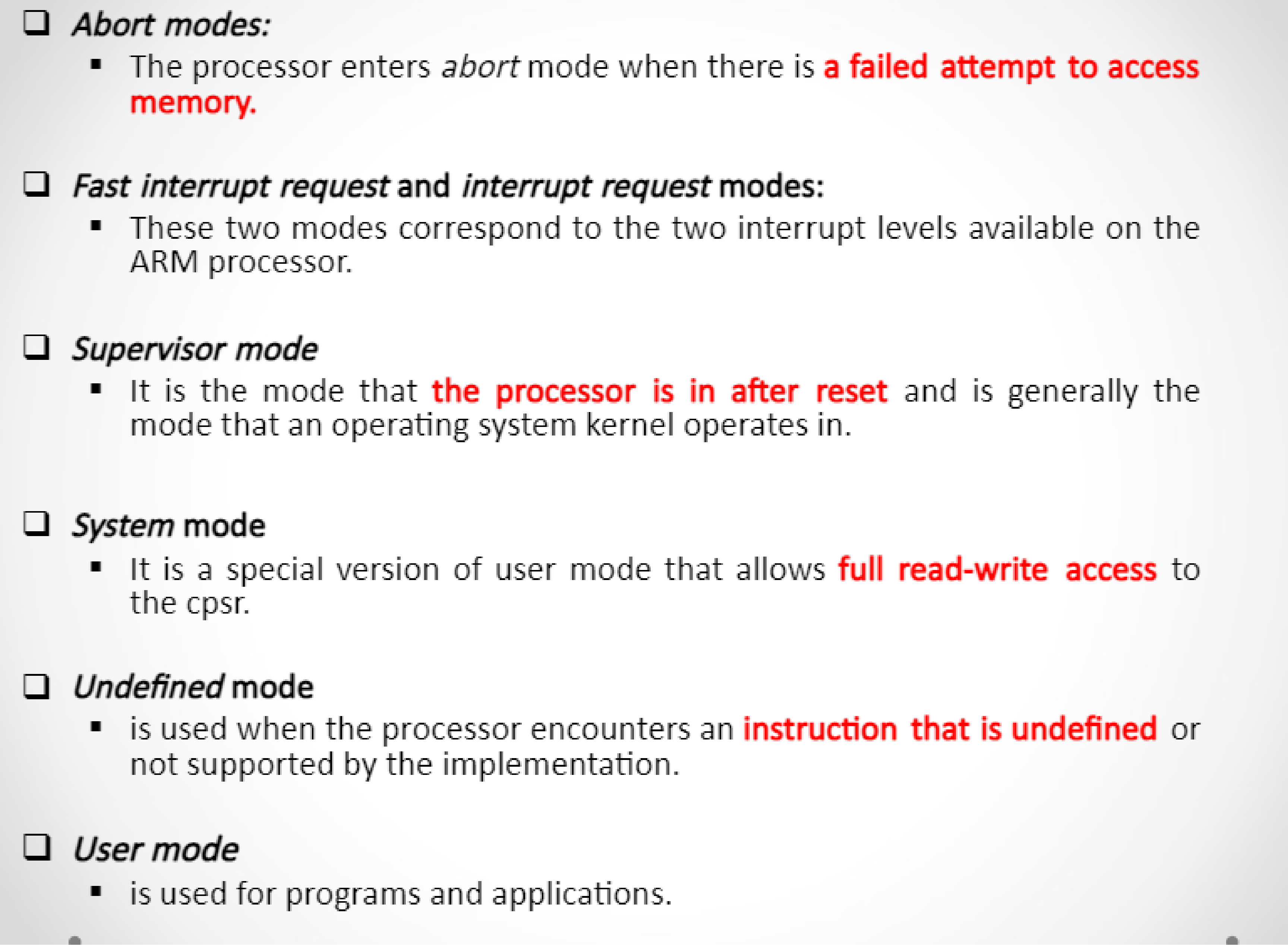
Processor Modes

- First 5 bits are signifies as mode selection

- The processor mode determines which registers are active and the access rights to the ***cpsr*** register itself.

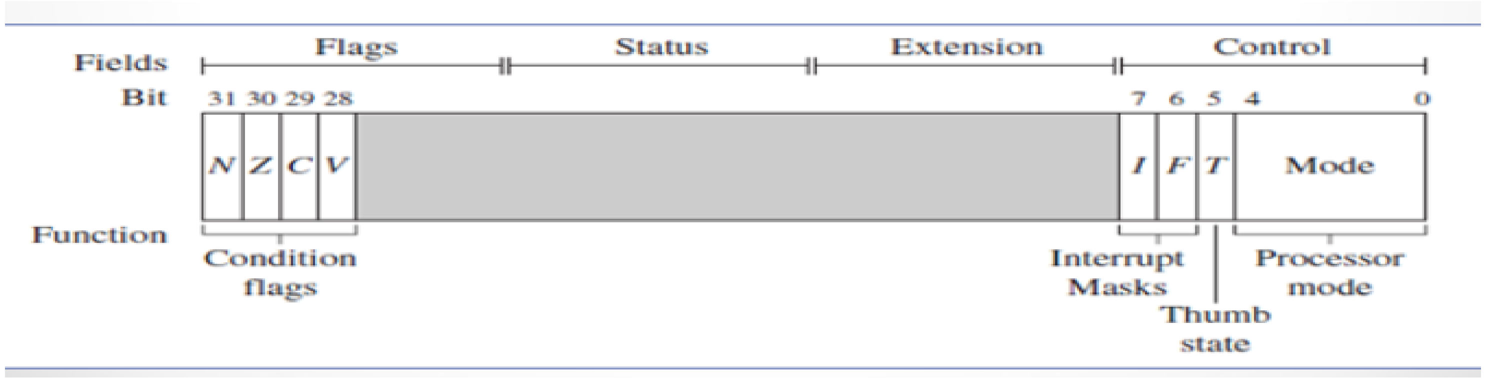
- Each processor mode is either **privileged** or **nonprivileged:** A **privileged** mode allows full read-write access to the **cpsr**. Conversely, a **nonprivileged** mode only allows read access to the control field in the *cpsr* but still allows read-write access to the condition flags.

- There are seven processor modes in total: **six privileged modes** (**abort**, **fast interrupt request**, **interrupt request**, **supervisor**, **system**, and **undefined**) and one **nonprivileged** mode (**user**).



- ***Abort modes:***
  - The processor enters *abort* mode when there is **a failed attempt to access memory.**
- ***Fast interrupt request* and *interrupt request* modes:**
  - These two modes correspond to the two interrupt levels available on the ARM processor.
- ***Supervisor mode***
  - It is the mode that **the processor is in after reset** and is generally the mode that an operating system kernel operates in.
- ***System* mode**
  - It is a special version of user mode that allows **full read-write access** to the cpsr.
- ***Undefined* mode**
  - is used when the processor encounters an **instruction that is undefined** or not supported by the implementation.
- ***User mode***
  - is used for programs and applications.

- The N bit is the "negative flag" and indicates that a value is negative.
  The Z bit is the "zero flag" and is set when an appropriate instruction produces a zero result.
  The C bit is the "carry flag" but it can also be used to indicate "borrows" (from subtraction operations)
- The V bit is the "overflow flag" which is set if an instruction produces a result that overflows and hence may go beyond the range of numbers that can be represented in 2's complement signed format.
- The I and F bits which determine whether interrupts (such as requests for input/output) are enabled or disabled.
- The T bit which indicates whether the processor is in "Thumb" mode, where the processor can execute a subset of the assembly language as 16-bit compact instructions. As Thumb code packs more instructions into the same amount of memory, it is an effective solution to applications where physical memory is at a premium.



Thank You